Recent Advancements in Tabular Deep Learning

Paper List

1. TabDDPM (ICML 2023)
2. STaSy (ICLR 2023)
3. CoDI (ICML 2023)
4. Revisiting Pretraining Objectives ~ (arxiv 2023)
5. TABR (ICLR 2024)
6. TabSyn (ICLR 2024)
7. TABM (ICLR 2025)
8. TabReD (ICLR 2025)
9. ModernNCA (ICLR 2025)
10. AnoLLM (ICLR 2025)
11. Latent Score-based Reweighting (ICLR 2025)
12. TabDiff (ICLR 2025)

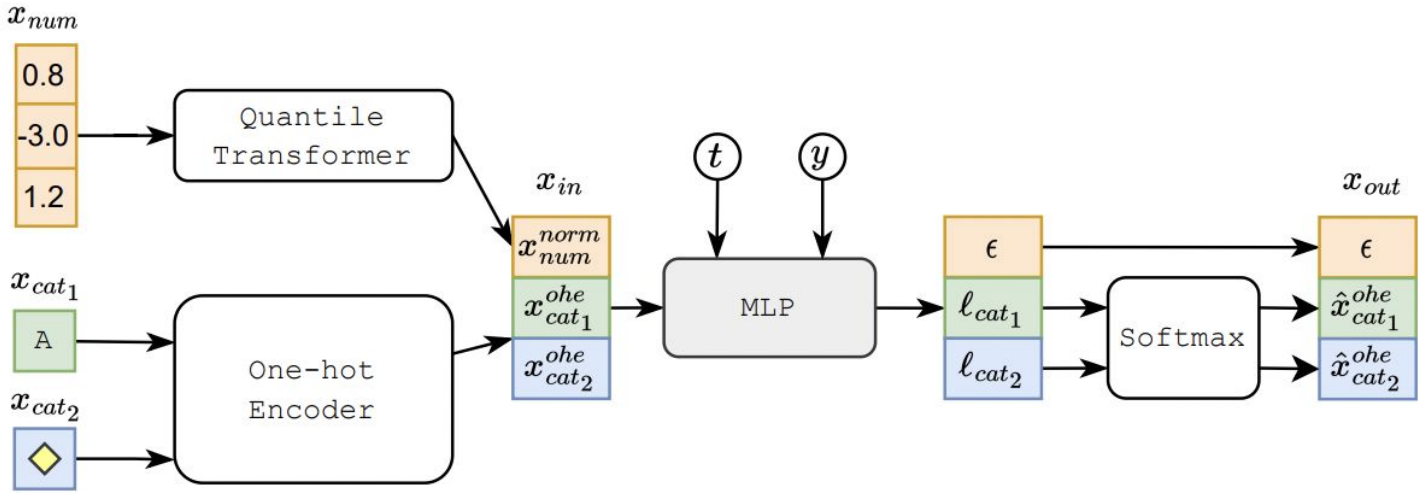
1. TabDDPM (ICML 2023)

(https://arxiv.org/pdf/2209.15421)

- **DDPM** for Tabular
- Tabular data = **Heterogeneous** features => Challenging
- **Numeric** & **Categorical**
  - Numeric: **Gaussian Quantile Transformation**
  - Categorical: Each categorical feature is **handled separately**
- Model: (Reverse step) **MLP**

1. TabDDPM (ICML 2023)

Figure 1. TabDDPM scheme for classification problems; t, y and ℓ denote a diffusion timestep, a class label, and logits, respectively.



1. TabDDPM (ICML 2023)

Table 4. The values of machine learning efficiency computed w.r.t. five weak classification/regression models. Negative scores denote negative R2, which means that performance is worse than an optimal constant prediction.

	AB ($R2$)	AD ($F1$)	BU ($F1$)	CA ($R2$)	CAR ($F1$)	CH ($F1$)	DE ($F1$)	DI ($F1$)
TVAE	$0.238_{\pm.012}$	$0.742_{\pm.001}$	$0.779_{\pm.004}$	$-13.0_{\pm1.51}$	$0.693_{\pm.002}$	$0.684_{\pm.003}$	$0.643_{\pm.003}$	$0.712_{\pm.010}$
CTABGAN	–	$0.737_{\pm.007}$	$0.786_{\pm.008}$	–	$0.684_{\pm.003}$	$0.636_{\pm.010}$	$0.614_{\pm.007}$	$0.655_{\pm.015}$
CTABGAN+	$0.316_{\pm.024}$	$0.730_{\pm.007}$	$0.837_{\pm.006}$	$-7.59_{\pm.645}$	$\mathbf{0.708_{\pm.002}}$	$0.650_{\pm.008}$	$0.648_{\pm.008}$	$\mathbf{0.727_{\pm.023}}$
SMOTE	$\mathbf{0.400_{\pm.009}}$	$0.750_{\pm.004}$	$0.842_{\pm.003}$	$0.667_{\pm.006}$	$0.693_{\pm.001}$	$0.690_{\pm.003}$	$0.649_{\pm.003}$	$0.677_{\pm.013}$
TabDDPM	$\mathbf{0.392_{\pm.009}}$	$\mathbf{0.758_{\pm.005}}$	$\mathbf{0.851_{\pm.003}}$	$\mathbf{0.695_{\pm.002}}$	$0.696_{\pm.001}$	$\mathbf{0.693_{\pm.003}}$	$\mathbf{0.659_{\pm.003}}$	$0.675_{\pm.011}$
Real	$0.423_{\pm.009}$	$0.750_{\pm.006}$	$0.845_{\pm.004}$	$0.663_{\pm.002}$	$0.683_{\pm.002}$	$0.679_{\pm.003}$	$0.648_{\pm.003}$	$0.699_{\pm.012}$

	FB ($R2$)	GE ($F1$)	HI ($F1$)	HO ($R2$)	IN ($R2$)	KI ($R2$)	MI ($F1$)	WI ($F1$)
TVAE	$\ll 0$	$0.372_{\pm.006}$	$0.590_{\pm.004}$	$0.174_{\pm.012}$	$0.470_{\pm.024}$	$0.666_{\pm.006}$	$\mathbf{0.880_{\pm.002}}$	$0.497_{\pm.001}$
CTABGAN	–	$0.339_{\pm.009}$	$0.539_{\pm.006}$	–	–	–	$0.856_{\pm.003}$	$0.656_{\pm.011}$
CTABGAN+	$\ll 0$	$0.373_{\pm.009}$	$0.598_{\pm.004}$	$0.222_{\pm.042}$	$0.669_{\pm.018}$	$0.197_{\pm.051}$	$0.867_{\pm.002}$	$0.653_{\pm.027}$
SMOTE	$\mathbf{0.651_{\pm.002}}$	$\mathbf{0.478_{\pm.005}}$	$0.664_{\pm.003}$	$0.394_{\pm.006}$	$0.709_{\pm.008}$	$\mathbf{0.751_{\pm.005}}$	$0.860_{\pm.001}$	$\mathbf{0.793_{\pm.004}}$
TabDDPM	$0.527_{\pm.005}$	$0.462_{\pm.005}$	$\mathbf{0.670_{\pm.002}}$	$\mathbf{0.426_{\pm.007}}$	$\mathbf{0.734_{\pm.007}}$	$0.611_{\pm.013}$	$0.850_{\pm.004}$	$\mathbf{0.792_{\pm.004}}$
Real	$0.645_{\pm.005}$	$0.431_{\pm.005}$	$0.663_{\pm.002}$	$0.415_{\pm.007}$	$0.708_{\pm.007}$	$0.768_{\pm.013}$	$0.850_{\pm.004}$	$0.684_{\pm.004}$

2. STaSy (ICLR 2023)

(https://arxiv.org/pdf/2210.04018)

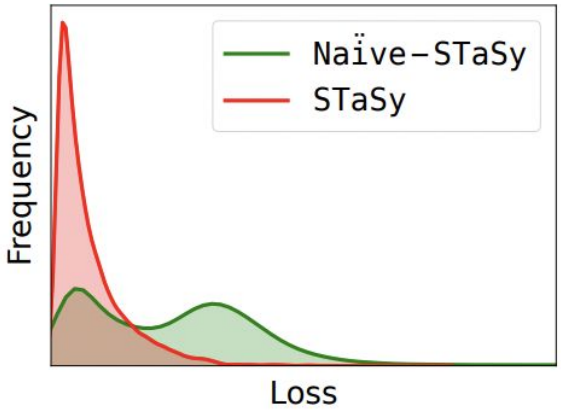


Figure 1: Distributions of denoising score matching loss in Shoppers

- Score-based Tabular data Synthesis (STaSy)
- **[1] Self-paced learning (SPL)**
  - In order to alleviate the **training difficulty**
  - Curriculum learning to select training records in a meaningful order (**easy -> hard**)
  - **"Learnable"** selection importance
- **[2] Fine-tuning strategy**
  - Reverse SDE process: Probability flow (Song et al.,) -> Exact likelihood calculation
  - Fine-tune based on the exact log-probability (**low prob samples = harder samples**)

2. STaSy (ICLR 2023)

Algorithm 1: How to train `STaSy`

Initialize $\boldsymbol{\theta}$, $\mathbf{v}$
`/* Train SGM based on our SPL training strategy */`
for *each mini-batch of records* **do**
  Update $\boldsymbol{\theta}$ after fixing $\mathbf{v}$ with Equation 7
  Update $\mathbf{v}$ with Equation 9
  Update α and β with the control method in Appendix D
`/* Fine-tune the trained model using log-probability */`
$\tau_i \leftarrow \log p(\mathbf{x}_i)$
$\mathcal{F} \leftarrow \{\mathbf{x}_i | \log p(\mathbf{x}_i)$, where $\mathbf{x}_i \in \mathcal{D}$, is smaller than the average (or median) log-probability.$\}$
for *each fine-tune epoch* **do**
  for *each* $\mathbf{x}_i \in \mathcal{F}$ **do**
    Update $\boldsymbol{\theta}$ with Equation 6
  $\mathcal{F} \leftarrow \{\mathbf{x}_i | \log p(\mathbf{x}_i) < \tau_i\}$
return $\boldsymbol{\theta}$

$$v_i^* = \begin{cases} 1, & \text{if } l_i \leq Q(\alpha), \\ 0, & \text{if } l_i \geq Q(\beta), \\ \dfrac{l_i - Q(\beta)}{Q(\alpha) - Q(\beta)}, & \text{otherwise.} \end{cases}$$

3. CoDI (ICML 2023)

(https://arxiv.org/pdf/2304.12654)

- Tabular **diffusion** model
- Difficulty in modeling **discrete** variables
- **[1] Architecture**
  - Process continuous and discrete variables **separately** by **two** diffusion models (but being **conditioned on each other**)
  - Two models are **co-evolved**
- **[2] CL**
  - To further bind the two diffusion models, introduce **CL with negative sampling**

3. CoDI (ICML 2023)

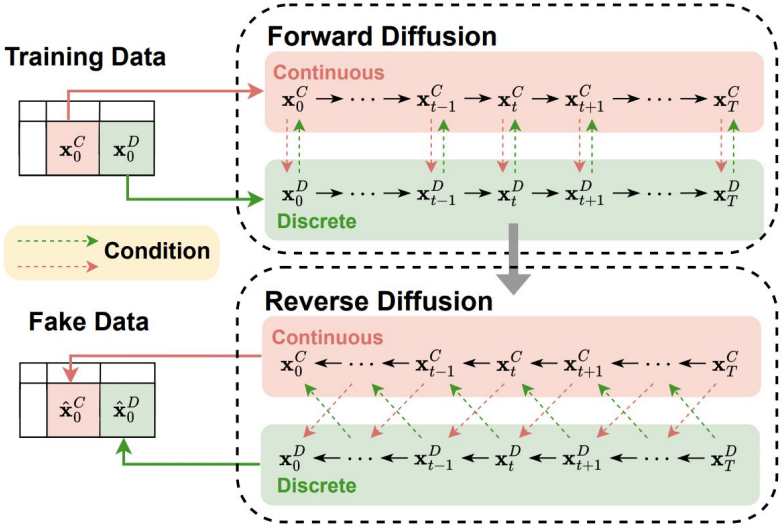


Figure 2: The overall workflow of co-evolving conditional diffusion models. Two diffusion models are connected by reading conditions from each other.

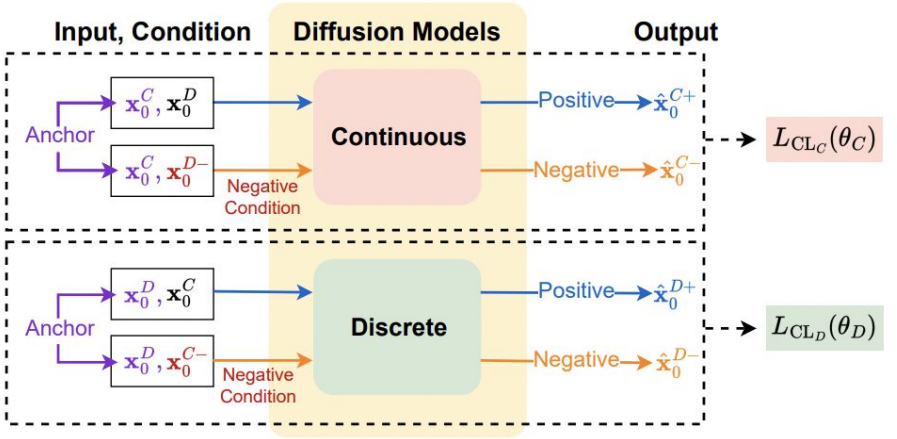


Figure 3: The proposed contrastive learning for tabular data. With a negative sampling method, we encourage the two diffusion models to generate samples that are closer to the positive samples and distinct from the negative samples.

3. CoDI (ICML 2023)

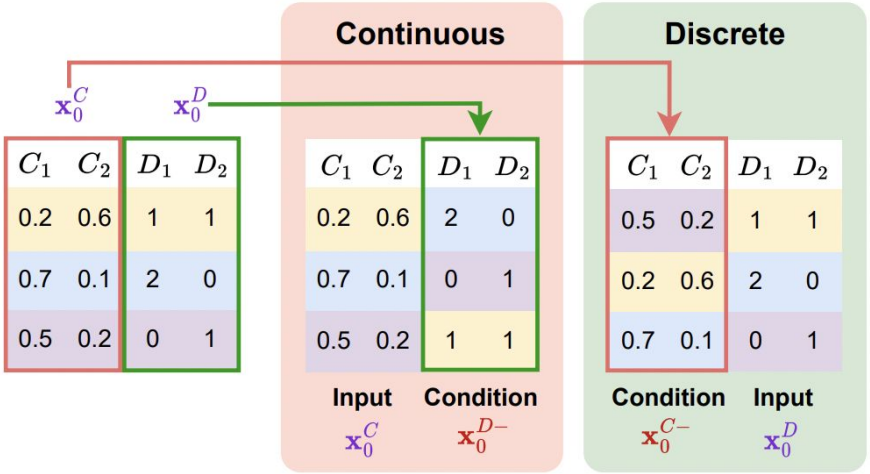


Figure 4: How to define the negative conditions. We randomly permute the continuous and discrete variable sets while maintaining their internal pairs and therefore, the inter-variable correlation does not make sense, i.e., they are not appropriate counterparts to each other.

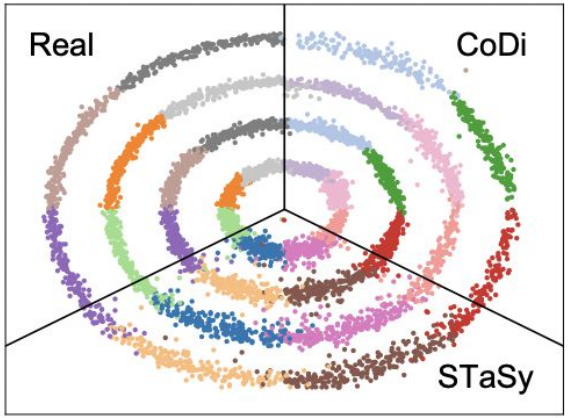


Figure 1: Preliminary experiment on a toy dataset. The dataset contains 4 columns, which are two continuous (the x-axis and y-axis) and two discrete (16 colors and 4 circles) columns. (Left) is a scatter plot of real data, (Bottom) is synthesized data by `STaSy` (Kim et al., 2022a), and (Right) is synthesized data by our proposed method. Detailed information and visualizations are in Appendix A.

3. CoDI (ICML 2023)

Algorithm 1 Training

Initialize θ_C and θ_D
repeat
  $\mathbf{x}_0^C \sim q(\mathbf{x}_0^C), \mathbf{x}_0^D \sim q(\mathbf{x}_0^D), t \sim \mathcal{U}(\{1, \dots, T\})$
  Compute $L_{\text{Diff}_\text{C}}(\theta_C)$ and $L_{\text{Diff}_\text{D}}(\theta_D)$
  Make negative conditions $\mathbf{x}_0^{C-}$ and $\mathbf{x}_0^{D-}$
  Generate positive samples $\hat{\mathbf{x}}_0^{C+}$ and $\hat{\mathbf{x}}_0^{D+}$
  Generate negative samples $\hat{\mathbf{x}}_0^{C-}$ and $\hat{\mathbf{x}}_0^{D-}$
  Compute $L_{\text{CL}_C}(\theta_C)$ and $L_{\text{CL}_D}(\theta_D)$
  $L_\text{C}(\theta_C) \leftarrow L_{\text{Diff}_\text{C}}(\theta_C) + \lambda_C L_{\text{CL}_\text{C}}(\theta_C)$
  $L_\text{D}(\theta_D) \leftarrow L_{\text{Diff}_\text{D}}(\theta_D) + \lambda_D L_{\text{CL}_\text{D}}(\theta_D)$
  Update θ_C and θ_D
until converged

Algorithm 2 Sampling

$\hat{\mathbf{x}}_T^C \sim p(\mathbf{x}_T^C), \hat{\mathbf{x}}_T^D \sim p(\mathbf{x}_T^D)$
for $i = T, \dots, 1$ **do**
  $\hat{\mathbf{x}}_{i-1}^C \sim p_{\theta_C}(\hat{\mathbf{x}}_{i-1}^C | \hat{\mathbf{x}}_i^C, \hat{\mathbf{x}}_i^D)$
  $\hat{\mathbf{x}}_{i-1}^D \sim p_{\theta_D}(\hat{\mathbf{x}}_{i-1}^D | \hat{\mathbf{x}}_i^D, \hat{\mathbf{x}}_i^C)$
end for
return $\hat{\mathbf{x}}_0^C, \hat{\mathbf{x}}_0^D$

4. Revisiting Pretraining Objectives ~ (arxiv 2023)

(https://arxiv.org/pdf/2207.03208)

- Unlike ML, **DL** can additionally **benefit from pretraining**!
  - Nonetheless, **not entirely clear** if pretraining provides consistent noticeable improvements in **tabular DL**!
- Goal: Aim to identify the **best practices to pretrain tabular DL models**
- Findings:
  - (1) Use the **object target labels** during the pretraining stage!
  - (2) **Properly performed pretraining** significantly increases the performance!

4. Revisiting Pretraining Objectives ~ (arxiv 2023)

- **Experiments 1) Pretraining objectives**
  - (1) Contrastive is not superior
  - (2) Pretraining is beneficial for the SoTA models
  - (3) No universal solution between self-prediction objectives (rec vs. mm)
- **Experiments 2) Target-aware pretraining objectives**
  - (1) Supervised loss with augmentations is another strong baseline for MLP
  - (2) Target-aware objectives demonstrate the best performance
- Summary: **(Standard) pretraining + Target-aware pretraining** is the best

4. Revisiting Pretraining Objectives ~ (arxiv 2023)

	GE ↑	CH ↑	CA ↓	HO ↓	OT ↓	HI ↑	FB ↓	AD ↑	WE ↓	CO ↑	MI ↓
MLP											
no pretraining	0.635	0.849	0.506	3.156	0.479	0.801	5.737	0.908	1.909	0.963	0.749
contrastive	0.672	**0.855**	0.455	**3.056**	0.469	**0.813**	5.697	0.910	1.881	0.960	0.748
rec	0.662	0.853	**0.445**	**3.044**	**0.466**	0.805	**5.641**	**0.910**	**1.875**	**0.965**	**0.746**
mask	**0.691**	**0.857**	0.454	3.113	0.472	**0.814**	**5.681**	**0.912**	1.883	0.964	0.748
MLP-PLR											
no pretraining	0.668	**0.858**	0.469	**3.008**	0.483	0.809	**5.608**	0.926	1.890	0.969	0.746
rec	0.667	0.852	**0.439**	**3.031**	**0.472**	0.808	**5.571**	**0.926**	**1.877**	**0.971**	**0.745**
mask	**0.685**	**0.863**	**0.434**	**3.007**	0.477	**0.818**	**5.586**	**0.927**	1.911	**0.970**	0.748
MLP-T-LR											
no pretraining	0.634	**0.866**	0.444	3.113	0.482	0.805	5.520	0.925	1.897	0.968	0.749
rec	**0.652**	0.857	**0.424**	3.109	**0.472**	0.808	**5.363**	0.924	**1.861**	**0.969**	**0.746**
mask	**0.654**	**0.868**	**0.424**	**3.045**	**0.472**	**0.818**	5.544	**0.926**	1.916	**0.969**	0.748

4. Revisiting Pretraining Objectives ~ (arxiv 2023)

	GE ↑	CH ↑	CA ↓	HO ↓	OT ↓	HI ↑	FB ↓	AD ↑	WE ↓	CO ↑	MI ↓	Avg. Rank
MLP												
no pretraining	0.635	0.849	0.506	3.156	0.479	0.801	5.737	0.908	1.909	0.963	0.749	5.5 ± 1.4
mask	0.691	0.857	0.454	3.113	0.472	0.814	5.681	0.912	1.883	0.964	0.748	3.8 ± 1.4
rec	0.662	0.853	0.445	**3.044**	0.466	0.805	5.641	0.910	1.875	0.965	0.746	3.6 ± 1.5
sup	0.693	0.856	0.441	3.077	**0.459**	0.814	5.689	0.914	1.883	0.968	0.748	3.0 ± 1.0
mask + target	0.683	0.857	0.434	**3.056**	0.468	**0.819**	5.633	0.914	1.876	0.965	0.748	2.9 ± 1.3
rec + target	0.659	0.853	0.454	**3.044**	0.463	0.806	5.636	0.909	1.884	0.965	**0.745**	3.7 ± 1.9
mask + sup	0.693	0.857	0.436	3.099	**0.458**	0.817	5.685	0.915	1.873	0.967	0.748	2.7 ± 1.2
rec + sup	0.684	0.854	0.436	**3.012**	**0.456**	0.815	5.672	0.911	**1.862**	0.967	0.747	2.6 ± 1.5
MLP-PLR												
no pretraining	0.668	0.858	0.469	**3.008**	0.483	0.809	5.608	0.926	1.890	0.969	0.746	3.5 ± 1.7
mask	0.685	**0.863**	0.434	**3.007**	0.477	0.818	5.586	0.927	1.911	0.970	0.748	2.8 ± 1.7
rec	0.667	0.852	0.439	**3.031**	0.472	0.808	5.571	0.926	1.877	**0.971**	**0.745**	2.6 ± 1.2
sup	**0.710**	0.859	0.433	3.136	0.479	0.811	5.521	0.924	1.873	**0.971**	0.748	2.5 ± 1.2
mask + target	0.694	0.862	**0.425**	**3.023**	0.474	**0.821**	5.537	**0.929**	1.911	0.969	0.749	2.5 ± 1.9
rec + target	0.688	0.860	0.445	**3.064**	0.475	0.812	5.507	0.927	1.887	**0.971**	0.748	2.7 ± 1.3
mask + sup	**0.711**	**0.866**	0.441	3.129	0.480	0.813	5.480	0.925	1.875	0.969	**0.745**	2.5 ± 1.4
rec + sup	**0.709**	0.858	0.433	**3.059**	0.465	0.807	5.571	0.927	**1.865**	**0.971**	**0.745**	1.9 ± 1.2
MLP-T-LR												
no pretraining	0.634	**0.866**	0.444	3.113	0.482	0.805	5.520	0.925	1.897	0.968	0.749	3.9 ± 1.7
mask	0.654	**0.868**	**0.424**	**3.045**	0.472	0.818	5.544	0.926	1.916	0.969	0.748	2.8 ± 1.7
rec	0.652	0.857	**0.424**	3.109	0.472	0.808	**5.363**	0.924	**1.861**	0.969	**0.746**	2.5 ± 1.4
sup	0.682	0.860	0.430	3.135	0.471	0.807	5.525	0.927	1.893	**0.971**	0.747	2.8 ± 1.5
mask + target	0.649	**0.865**	**0.421**	**3.058**	0.474	**0.820**	5.644	**0.929**	1.924	0.969	0.749	2.8 ± 2.1
rec + target	0.668	**0.864**	0.440	3.113	0.473	0.806	5.493	0.927	**1.862**	0.969	**0.746**	2.5 ± 1.4
mask + sup	0.676	0.858	0.429	3.199	0.468	0.814	5.510	0.926	1.869	**0.971**	0.748	2.5 ± 0.8
rec + sup	0.678	**0.865**	0.437	3.112	0.462	0.807	5.516	0.927	**1.862**	**0.970**	0.748	2.4 ± 1.2

5. TABR (ICLR 2024)

(https://arxiv.org/pdf/2307.14338)

- Non-DL algorithms based on **GBDT**: Strong baseline
- **TabR = Retrieval-based tabular DL model = FFN + kNN**
- Incremental approach:
  - Step 0) The vanilla-attention-like baseline
  - Step 1) Adding context labels
  - Step 2) Improving the similarity module
  - Step 3) Improving the value module
  - Step 4) TabR
- Outperforms GBDT models on the recently proposed “GBDT-friendly” benchmark

5. TABR (ICLR 2024)

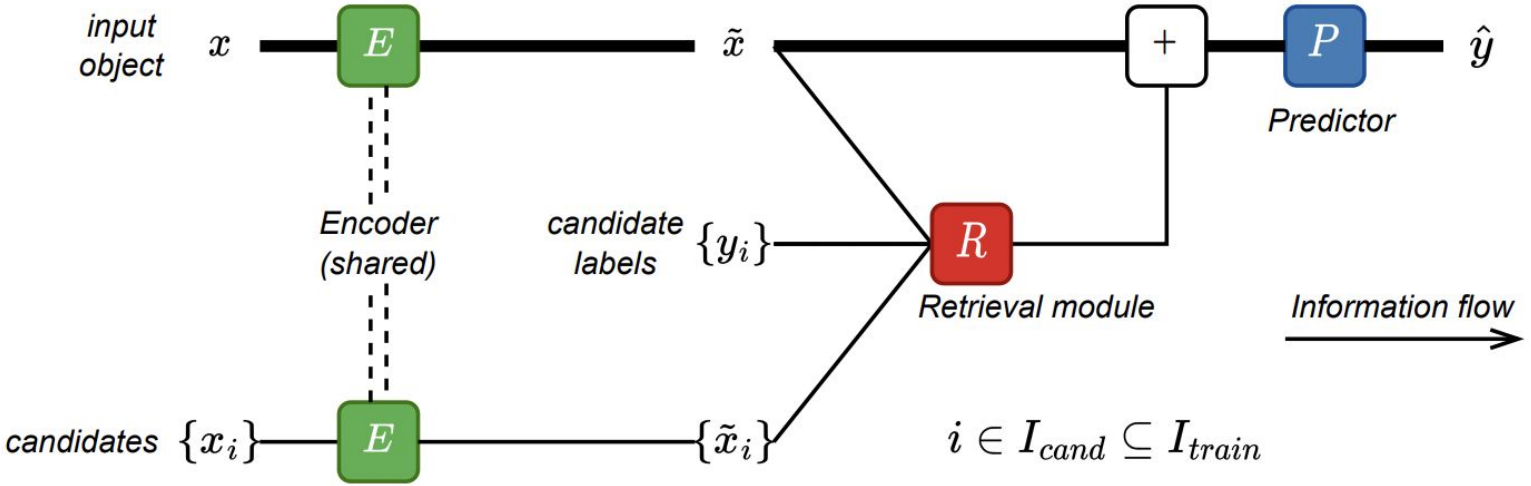


Figure 2: The generic retrieval-based architecture introduced in subsection 3.2 and used to build TabR. First, a target object and its candidates for retrieval are encoded with the same encoder E. Then, the retrieval module R enriches the target object's representation by retrieving and processing relevant objects from the candidates. Finally, predictor P makes a prediction. The bold path highlights the structure of the feed-forward retrieval-free model before the addition of the retrieval module R.

5. TABR (ICLR 2024)

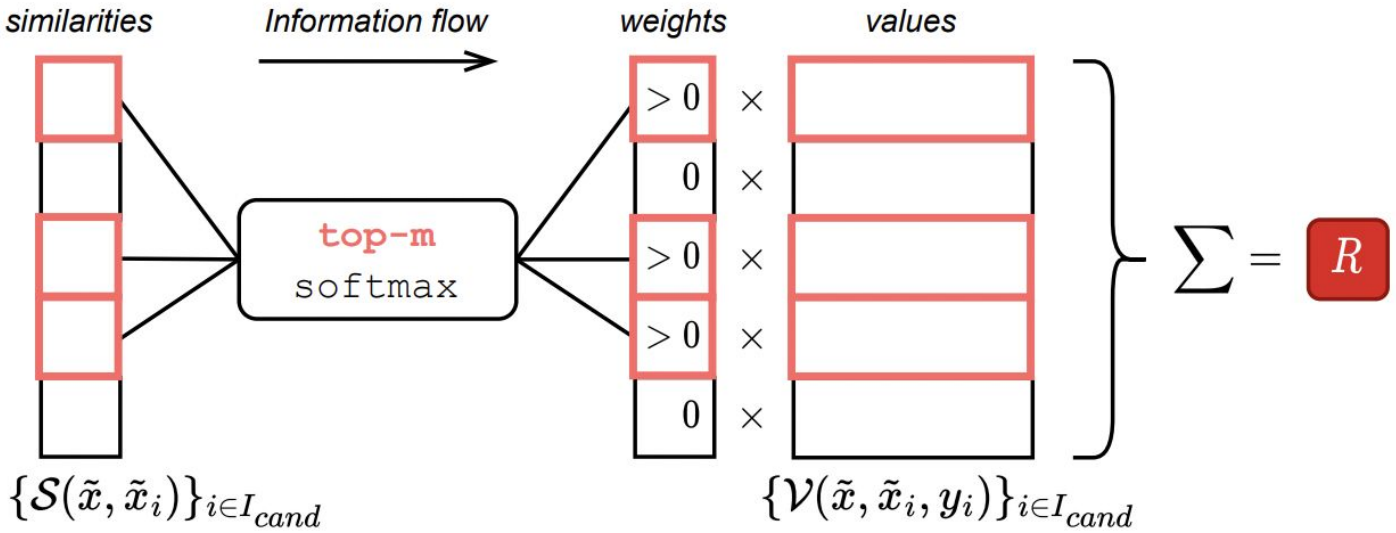


Figure 4: Simplified illustration of the retrieval module R introduced in Figure 2 (the omitted details are provided in the main text). For the target object's representation $\tilde{x}$, the module takes the m nearest neighbors among the candidates $\{\tilde{x}_i\}$ according to the similarity module $S : (\mathbb{R}^d, \mathbb{R}^d) \to \mathbb{R}$ and aggregates their values produced by the value module $\mathcal{V} : (\mathbb{R}^d, \mathbb{R}^d, \mathbb{Y}) \to \mathbb{R}^d$.

6. TabSyn (ICLR 2024)

(https://arxiv.org/pdf/2310.09656)

- **Synthesizes** tabular data by leveraging a **diffusion model** within a **VAE latent space**
  - (1) Diffusion = **Latent diffusion model (LDM)**
  - (2) **VAE** = Tokenizer + Transformer encoder + Transformer decoder + Detokenizer
- Advantages
  - (1) **Generality**: Broad spectrum of **data types** => Into a single unified space
  - (2) **Quality**: Optimizing the distribution of **latent embeddings**
  - (3) **Speed**: Much **fewer** number of reverse steps

6. TabSyn (ICLR 2024)

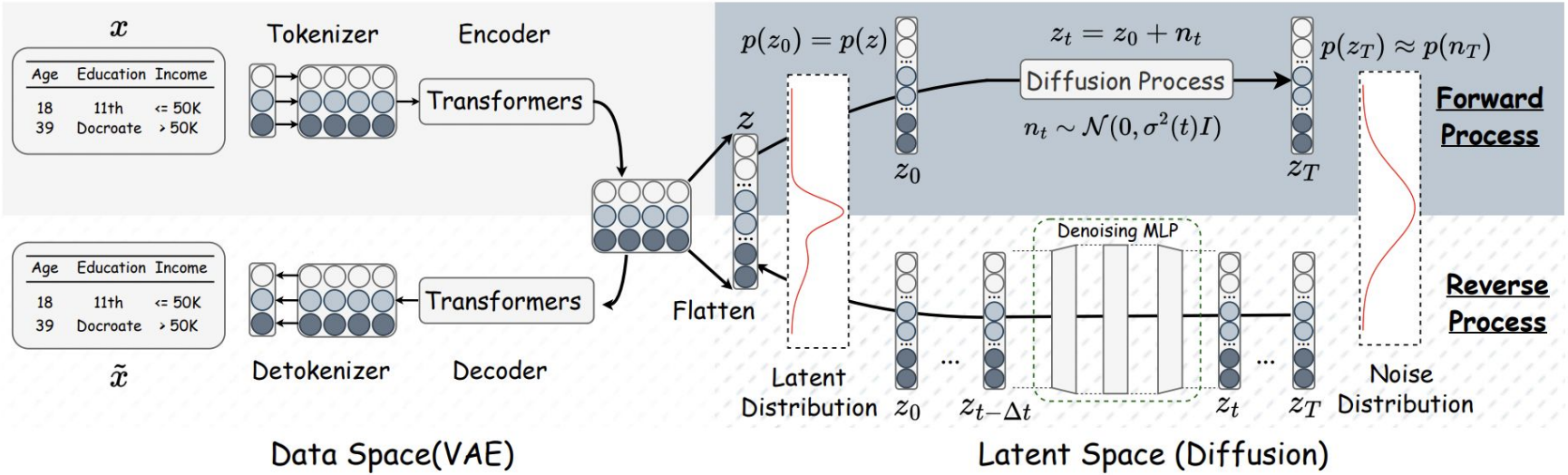


Figure 2: An overview of the proposed TABSYN. Each row data x is mapped to latent space z via a column-wise tokenizer and an encoder. A diffusion process $z_0 \rightarrow z_T$ is applied in the latent space. Synthesis $z_T \rightarrow z_0$ starts from the base distribution $p(z_T)$ and generates samples z_0 in latent space through a reverse process. These samples are then mapped from latent z to data space $\tilde{x}$ using a decoder and a detokenizer.

7. TABM (ICLR 2025)

(https://arxiv.org/pdf/2410.24210)

- Designing substantially better **MLP-based** tabular archs
- TabM = **Efficient ensembling** (of MLPs)
  - Produces multiple predictions per object
  - Parameter-efficient deep “ensembles”.
  - Three versions
    - MLP + **Packed-Ensemble**
    - MLP + **BatchEnsemble**
    - MLP + **MiniEnsemble**

7. TABM (ICLR 2025)

- vs. Traditional deep ensemble
  - MLPs are trained **simultaneously** & **share** most of their params
- Experiments:
  - Large-scale evaluation of tabular DL architectures on public benchmarks
  - Task performance and efficiency
- **Stronger and more practical models** (vs. to attention- and retrieval based archs)

7. TABM (ICLR 2025)

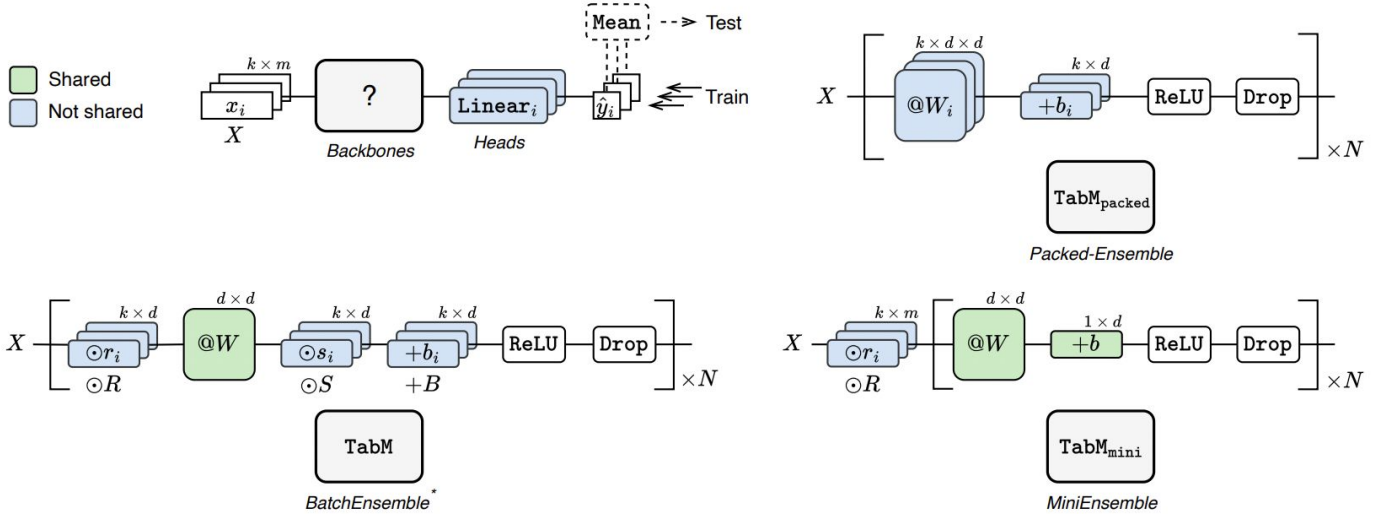


Figure 1: *(Upper left)* A high-level illustration of TabM. One TabM represents an ensemble of k MLPs processing k inputs in parallel. The remaining parts of the figure are three different parametrizations of the k MLP backbones. *(Upper right)* $\text{TabM}_{\text{packed}}$ consists of k fully independent MLPs. *(Lower left)* TabM is obtained by injecting three non-shared adapters R, S, B in each of the N linear layers of *one* MLP (* the initialization differs from Wen et al. (2020)). *(Lower right)* $\text{TabM}_{\text{mini}}$ is obtained by keeping only the very first adapter R of TabM and removing the remaining $3N - 1$ adapters. *(Details)* Input transformations such as one-hot-encoding or feature embeddings (Gorishniy et al., 2022) are omitted for simplicity. `Drop` denotes dropout (Srivastava et al., 2014).

8. TabReD (ICLR 2025)

(https://arxiv.org/pdf/2406.19380)

- Real-world tabular data:
  - Time-based distribution shifts
  - Complex feature engineering pipelines
- Existing benchmarks lack:
  - Timestamps for temporal splits
  - Realistic feature construction
- **TabReD benchmark**:
  - (1) 8 industry-grade tabular datasets / (2) Includes timestamps
  - (3) Reflects real-world feature engineering / (4) Avoids data leakage

8. TabReD (ICLR 2025)

Table 1: The landscape of existing tabular machine learning benchmarks compared to TabReD. We report median dataset sizes, number of features, the number of datasets with various issues. The "Time-splits" column is reported only for the datasets without issues. We see that the datasets semi-automatically gathered from OpenML (Tabzilla and Grinsztajn et al. (2022)) contain more quality issues. Furthermore, no benchmark besides TabReD focuses on temporal-shift based evaluation and less than half of datasets in each benchmark have timestamp metadata needed for time-based validation availability.

* – the original dataset, introduced in (Malinin et al., 2021) has the canonical OOD split, but the standard IID split commonly used contains time-based leakage.

** – the median full dataset size. In experiments, to reduce compute requirements, we use subsampled versions of the TabReD datasets.

Benchmark	Dataset Sizes (Q_{50})		Issues (#Issues / #Datasets)			Time-split		
	#Samples	#Features	Data-Leakage	Synthetic or Untraceable	Non-Tabular	Needed	Possible	Used
Grinsztajn et al. (2022)	16,679	13	7 / 44	1 / 44	7 / 44	22	5	
Tabzilla (McElfresh et al., 2023)	3,087	23	3 / 36	6 / 36	12 / 36	12	0	
WildTab (Kolesnikov, 2023)	546,543	10	1* / 3	1 / 3	0 / 3	1	1	✗
TableShift (Gardner et al., 2023)	840,582	23	0 / 15	0 / 15	0 / 15	15	8	
Gorishniy et al. (2024)	57,909	20	1* / 10	1 / 10	0 / 10	7	1	
TabReD (ours)	7,163,150**	261	✗	✗	✗	✓	✓	✓

9. ModernNCA (ICLR 2025)

(https://arxiv.org/pdf/2407.03257)

- Can **classical methods (e.g., kNN)** can be **revitalized** with modern techniques?
- Revisit a differentiable version of kNN = **Neighbourhood Components Analysis (NCA)**
  - Learn a **linear projection** to capture semantic similarities between instances
  - Add **modern DL** on top
- **NCA using SGD** (w/o dimensionality reduction) = Decent performance!
- Analyzing the factors behind these improvements
  - e.g., loss functions, prediction strategies, and deep architectures…

9. ModernNCA (ICLR 2025)

- **(1) Learning objectives**
  - Classification: **Soft-NN loss**
    i. Predicting the label of a target instance
    ii. By computing a **weighted average of its neighbors** across the C class
  - Regression: Weighted sum of scalar labels from the neighborhood
- **(2) Prediction**
  - Traditional “hard” KNN approach (X)
  - Adopt the **soft-NN rule (O)** => Applicable to **both Cls & Reg**
- **(3) Arch**
  - **NCA + non-linear layers**

9. ModernNCA (ICLR 2025)

- **(4) Stochastic Neighborhood Sampling**
  - Training: **Subset of the training set** is randomly sampled for each mini-batch
  - Inference: Searches for neighbors using the **entire training set**
- **(5) Distance function**
  - **Euclidean** distance

Learning Objective. Assume the label y_j is continuous in regression tasks and in one-hot form for classification tasks. We modify Equation 3 as follows:

$$\hat{y}_i = \sum_{(\boldsymbol{x}_j, y_j) \in \mathcal{D}} \frac{\exp\left(-\operatorname{dist}^2(\phi(\boldsymbol{x}_i),\ \phi(\boldsymbol{x}_j))\right)}{\sum_{(\boldsymbol{x}_l, y_l) \in \mathcal{D}, \boldsymbol{x}_l \neq \boldsymbol{x}_i} \exp\left(-\operatorname{dist}^2(\phi(\boldsymbol{x}_i),\ \phi(\boldsymbol{x}_l))\right)} y_j \ . \tag{4}$$

This formulation ensures that similar instances (based on their distance in the embedding space mapped by ϕ) yield closer predictions. For classification, Equation 4 generalizes Equation 3, pre-

9. ModernNCA (ICLR 2025)

Neighbourhood Component Analysis (NCA). NCA focuses on the classification task (Goldberger et al., 2004). According to the 1NN rule, NCA defines the probability that $\boldsymbol{x}_j$ locates in the neighborhood of $\boldsymbol{x}_i$ by

$$\Pr(\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i; \mathcal{D}) \mid \boldsymbol{x}_i, \mathcal{D}, \boldsymbol{L}) = \frac{\exp\left(-\operatorname{dist}^2(\boldsymbol{L}^\top \boldsymbol{x}_i,\ \boldsymbol{L}^\top \boldsymbol{x}_j)\right)}{\sum_{(\boldsymbol{x}_l, y_l) \in \mathcal{D}, \boldsymbol{x}_l \neq \boldsymbol{x}_i} \exp\left(-\operatorname{dist}^2(\boldsymbol{L}^\top \boldsymbol{x}_i,\ \boldsymbol{L}^\top \boldsymbol{x}_l)\right)} . \tag{2}$$

Then, the posterior probability that an instance $\boldsymbol{x}_i$ is classified as the class y_i is:

$$\Pr(\hat{y}_i = y_i \mid \boldsymbol{x}_i, \mathcal{D}, \boldsymbol{L}) = \sum_{(\boldsymbol{x}_j, y_j) \in \mathcal{D} \wedge y_j = y_i} \Pr(\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i; \mathcal{D}) \mid \boldsymbol{x}_i, \mathcal{D}, \boldsymbol{L}) . \tag{3}$$

$\boldsymbol{L} \in \mathbb{R}^{d \times d'}$ is a linear projection usually with $d' \leq d$, which reduces the dimension of the raw input. Therefore, the posterior that an instance $\boldsymbol{x}_i$ belongs to the class y_i depends on its similarity (measured by the negative squared Euclidean distance in the space projected by $\boldsymbol{L}$) between its neighbors from

9. ModernNCA (ICLR 2025)

Neighbourhood Component Analysis (NCA). NCA focuses on the classification task (Goldberger et al., 2004). According to the 1NN rule, NCA defines the probability that $\boldsymbol{x}_j$ locates in the neighborhood of $\boldsymbol{x}_i$ by

$$\Pr(\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i; \mathcal{D}) \mid \boldsymbol{x}_i, \mathcal{D}, \boldsymbol{L}) = \frac{\exp\left(-\operatorname{dist}^2(\boldsymbol{L}^\top \boldsymbol{x}_i,\ \boldsymbol{L}^\top \boldsymbol{x}_j)\right)}{\sum_{(\boldsymbol{x}_l, y_l) \in \mathcal{D}, \boldsymbol{x}_l \neq \boldsymbol{x}_i} \exp\left(-\operatorname{dist}^2(\boldsymbol{L}^\top \boldsymbol{x}_i,\ \boldsymbol{L}^\top \boldsymbol{x}_l)\right)} . \tag{2}$$

Then, the posterior probability that an instance $\boldsymbol{x}_i$ is classified as the class y_i is:

$$\Pr(\hat{y}_i = y_i \mid \boldsymbol{x}_i, \mathcal{D}, \boldsymbol{L}) = \sum_{(\boldsymbol{x}_j, y_j) \in \mathcal{D} \wedge y_j = y_i} \Pr(\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i; \mathcal{D}) \mid \boldsymbol{x}_i, \mathcal{D}, \boldsymbol{L}) . \tag{3}$$

$\boldsymbol{L} \in \mathbb{R}^{d \times d'}$ is a linear projection usually with $d' \leq d$, which reduces the dimension of the raw input. Therefore, the posterior that an instance $\boldsymbol{x}_i$ belongs to the class y_i depends on its similarity (measured by the negative squared Euclidean distance in the space projected by $\boldsymbol{L}$) between its neighbors from

10. AnoLLM (ICLR 2025)

(https://openreview.net/pdf?id=7VkHffT5X2)

- Leverage **LLMs** for **unsupervised tabular AD**
- Convert tabular -> standardized **text format**
- Adapt a pre-trained LLM with this serialized data
- **Fine-tuning a pretrained LLM** with serialized tabular data (feat. **language modelling** loss)

10. AnoLLM (ICLR 2025)

- Assign **anomaly scores** based on the **NLL**
- Pros)
  - Preserves data integrity & Streamlines the preprocessing required for tabular AD
  - Effectively handle **mixed-type** data (Especially those containing textual features)
- During **inference**)
  - Anomaly scores are determined by averaging the **NLL across r random permutations** of the test data.

10. AnoLLM (ICLR 2025)

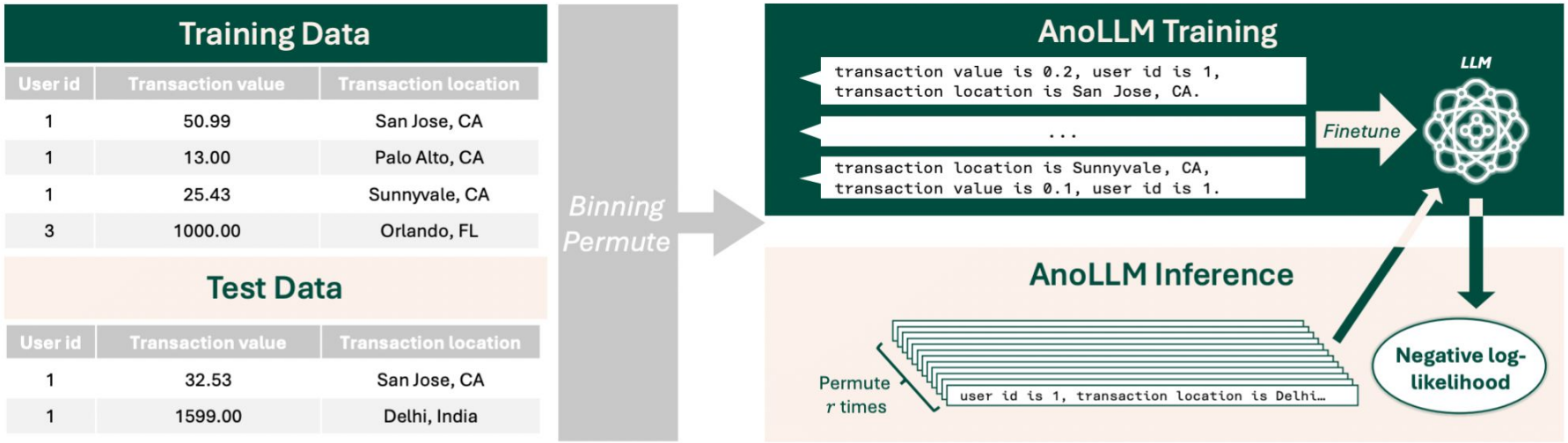


Figure 1: Overall framework of AnoLLM. During the preprocessing stage, numerical columns are binned into groups, and each data row is transformed into a natural language sequence with a randomly shuffled order of columns. In the training stage, a pretrained LLM is fine-tuned using the preprocessed tabular data. During inference, anomaly scores are determined by averaging the negative log-likelihood across r random permutations of the test data.

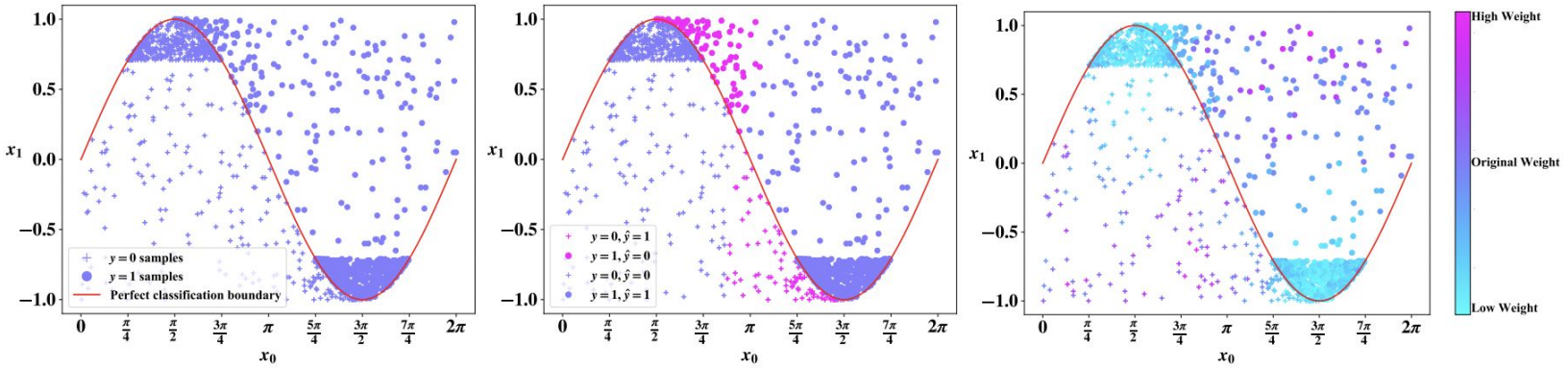
11. Latent Score-based Reweighting (ICLR 2025)

(https://openreview.net/pdf?id=HSLClc1a7W)

- ML models = Underperform on specific subsets
  - Due to **inherent biases** and spurious correlations in the training data!
- Proposal: **Latent score-based reweighting framework**
  - Leverages **score-based models** to capture the **joint data distribution p(x,y)**
  - Estimate **sample density** through the **similarity** of score vectors with neighbor
  - Identifies underrepresented regions and **upweights samples** accordingly!
- Results: Directly tackles inherent **data imbalances**

  => Enhancing robustness by ensuring a more **uniform dataset representation**
- Loss function: **Weighted** CE loss

11. Latent Score-based Reweighting (ICLR 2025)



(a) Unweighted training data (b) Weighted data by JTT (c) Weighted data by latent score

$$w_i = \frac{\exp(-\text{SimDiff}(z_i)/\tau)}{\sum_{j=0}^{N-1} \exp(-\text{SimDiff}(z_j)/\tau)}, \tag{13}$$

where N is the number of all training samples and τ denotes a temperature which controls the scale of reweighting. Finally, we train an unbiased classification model ψ as:

$$\mathcal{L}_{\text{classification}} = \mathbb{E}_{(z_i, y_i)}[w_i \ell(\psi(z_i), y_i)], \tag{14}$$

where ℓ stands for cross entropy loss. When testing, we only use ϕ_{Enc} and ψ to make predictions.

12. TabDiff (ICLR 2025)

(https://arxiv.org/pdf/2410.20626)

- Generative models for tabular data: Challenging due to ...
  - Heterogeneous data types
  - Complex inter-correlations
  - Intricate column-wise distributions
- Proposal: **TabDiff**
  - **Joint** diffusion framework that models all **mixed-type distributions** of tabular data in a **single model**

12. TabDiff (ICLR 2025)

(https://arxiv.org/pdf/2410.20626)

- Key innovation: **Joint continuous-time diffusion process**
  (for **numerical** and **categorical**)
  - Feature-wise learnable diffusion processes to counter the high disparity of different feature distributions.
  - Parameterized by a transformer handling different input types
- Others:
  - **Mixed-type stochastic sampler**
    - To automatically correct the accumulated decoding error during sampling
  - **Classifier-free guidance** for conditional missing column value imputation

12. TabDiff (ICLR 2025)

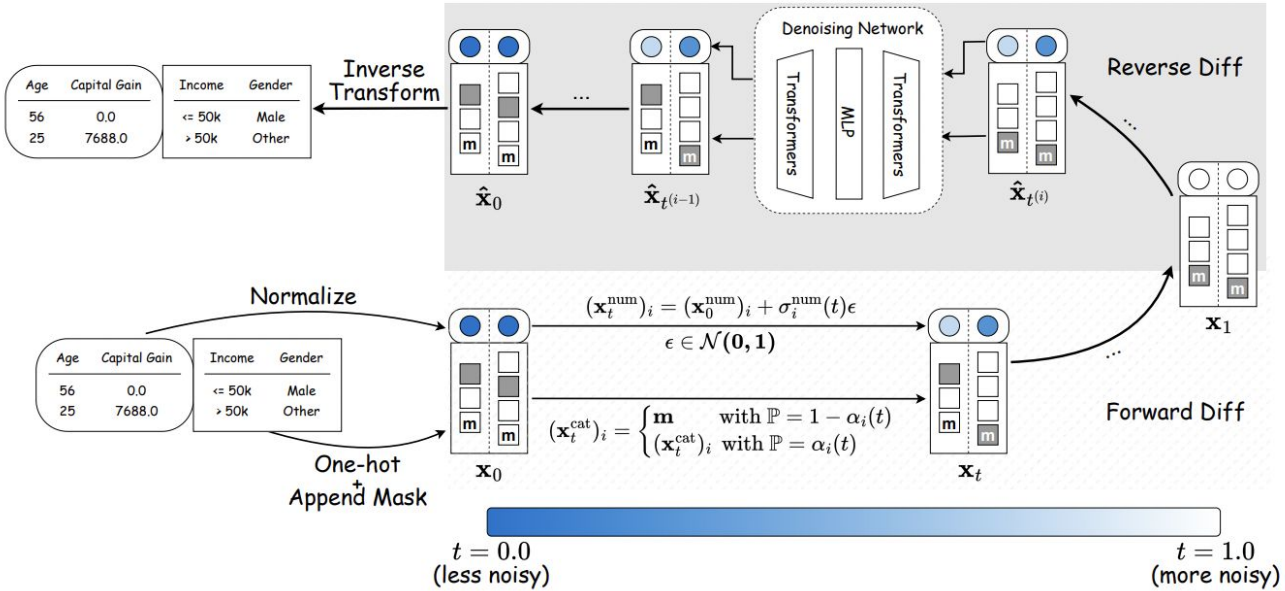


Figure 1: A high-level overview of TABDIFF. TABDIFF operates by normalizing numerical columns and converting categorical columns into one-hot vectors with an extra [MASK] class. Joint forward diffusion processes are applied to all modalities with each column's noise rate controlled by learnable schedules. New samples are generated via reverse process, with the denoising network gradually denoising $\mathbf{x}_1$ into $\mathbf{x}_0$ and then applying the inverse transform to recover the original format.

12. TabDiff (ICLR 2025)

Algorithm 1 Training

repeat
  Sample $\mathbf{x}_0 \sim p_0(\mathbf{x})$
  Sample $t \sim U(0,1)$
  Sample $\boldsymbol{\epsilon}_{\text{num}} \sim \mathcal{N}(0, \mathbf{I}_{M\,\text{num}})$
  $\mathbf{x}_t^{\text{num}} = \mathbf{x}_0^{\text{num}} + \boldsymbol{\sigma}^{\text{num}}(t)\boldsymbol{\epsilon}_{\text{num}}$
  $\boldsymbol{\alpha}_t = \exp(-\boldsymbol{\sigma}^{\text{cat}}(t))$
  Sample $\mathbf{x}_t^{\text{cat}} \sim \mathrm{q}(\mathbf{x}_t|\mathbf{x}_0, \boldsymbol{\alpha}_t)$ Eq. (6)
  $\mathbf{x}_t = [\mathbf{x}_t^{\text{num}}, \mathbf{x}_t^{\text{cat}}]$
  Take gradient descent step on $\nabla_{\theta,\rho,k}\mathcal{L}_{\text{TABDIFF}}$
until converged

Algorithm 2 Sampling

Sample $\mathbf{x}_T^{\text{num}} \sim \mathcal{N}(0, \mathbf{I}_{M\,\text{num}})$, $\mathbf{x}_T^{\text{cat}} = \boldsymbol{m}$
for $t = T$ to 1 **do**
  $t^+ \leftarrow t + \gamma_t t, \gamma_t = 1/T$
  ▷ Numerical forward perturbation:
  Sample $\boldsymbol{\epsilon}^{\text{num}} \sim \mathcal{N}(0, \mathbf{I}_{M\,\text{num}})$
  $\mathbf{x}_{t^+}^{\text{num}} \leftarrow \mathbf{x}_t^{\text{num}} + \sqrt{\boldsymbol{\sigma}^{\text{num}}(t^+)^2 - \boldsymbol{\sigma}^{\text{num}}(t)^2}\boldsymbol{\epsilon}^{\text{num}}$
  ▷ Categorical forward perturbation:
  Sample $\mathbf{x}_{t^+}^{\text{cat}} \sim q\left(\mathbf{x}_{t^+}^{\text{cat}}|\mathbf{x}_t^{\text{cat}}, 1 - \boldsymbol{\alpha}_{t^+}/\boldsymbol{\alpha}_t\right)$ Eq. (6)
  ▷ Concatenate:
  $\mathbf{x}_{t^+} = [\mathbf{x}_{t^+}^{\text{num}}, \mathbf{x}_{t^+}^{\text{cat}}]$
  ▷ Numerical backward ODE:
  $d\mathbf{x}^{\text{num}} = (\mathbf{x}_{t^+}^{\text{num}} - \mu_\theta^{\text{num}}(\mathbf{x}_{t^+}, t^+))/\boldsymbol{\sigma}^{\text{num}}(t^+)$
  $\mathbf{x}_{t-1}^{\text{num}} \leftarrow \mathbf{x}_{t^+}^{\text{num}} + (\boldsymbol{\sigma}^{\text{num}}(t-1) - \boldsymbol{\sigma}^{\text{num}}(t^+))d\mathbf{x}^{\text{num}}$
  ▷ Categorical backward sampling:
  Sample $\mathbf{x}_{t-1}^{\text{cat}} \sim p_\theta(\mathbf{x}_{t-1}^{\text{cat}}|\mathbf{x}_{t^+}^{\text{cat}}, \mu_\theta^{\text{cat}}(\mathbf{x}_{t^+}, t^+))$ Eq. (8)
end for
return $\mathbf{x}_0^{\text{num}}, \mathbf{x}_0^{\text{cat}}$